第31卷 第19期
2009年10月

武 汉 理 工 大 学 学 报
JOURNAL OF WUHAN UNIVERSITY OF TECHNOLOGY

Vol.31 No.19
Oct. 2009

DOI: 10.3963/j.issn.1671-4431.2009.19.037

基于嵌入式ARM的音频WMA定点优化

苏 扬, 胡绍江, 郭 倩
(武汉理工大学信息工程学院, 武汉 430070)

摘 要: 为提升ARM上WMA解码性能, 基于FFMPEG工程的解码算法, 简要地介绍了浮点转定点、快速除法处理、汇编化、特殊函数算法优化等4个优化方面, 重点描述了浮点存储格式及浮点数转定点数的方法, 此优化的结果使WMA的解码速度获得大幅度提高, 并在I.MX21(ARM926内核)平台上得到验证。

关键词: WMA; IEEE754标准; ARM926; MDCT; 汇编指令; FFMPEG
中图分类号: TP 311.1; TP 37 文献标识码: A 文章编号: 1671-4431(2009)19-0137-04

Integer Optimization of WMA Audio Based on Embedded ARM Environment

SU Yang, HU Shao-jiang, GUO Qian
(School of Information Engineering, Wuhan University of Technology, Wuhan 430070, China)

Abstract: For the better performance of WMA decoding on ARM Environment, four optimizations of WMA decoding based on FFMPEG, including principle of Float-point to Integer-point, Express division, Assemble and Algorithm of special functions optimization are introduced. It focuses on the description of the conversion between Float-point and Integer-point. The decoding had a good performance with the help of optimizations, and the result was verified on I.MX21 (ARM926 Core) platform.

Key words: WMA; IEEE754; ARM926; MDCT; assemble directive; FFMPEG

在目前移动平台多媒体领域上, WMA编码的音频文件以较高的压缩率(与MP3相比, 同样的音质播放码流更小)且较好的音质应用广泛。但WMA解码算法(MDCT离散余弦变换)相对复杂, 解码速度跟设备的芯片性能联系紧密, 若芯片性能不够, 其结果是声音会出现失真。其中一种经济可行的解决方法是在原来的硬件平台上改善软件解码算法, 结合硬件的独有特点, 最大限度地挖掘硬件的潜能, 就可使音视频播放性能获得大幅度地提升。因此, 对于低成本、低功耗、高灵活度的嵌入式应用, 倾向于采用经过优化过的软件解码方案。现在流行的FFMPEG工程下的WMA解码算法只适合于PC机或高端处理器PDA上使用, 在I.MX21平台上(ARM926内核)运行的效果不理想[1], 声音慢且失真现象严重, 但经过浮点转定点、快速除法处理、汇编化及MDCT解码阶段特殊函数算法优化这4个方面的改进后[2,3], 同样的条件下, 文件播放流畅, 声音正常。

1 WMA解码优化的研究

1.1 浮点转定点分析

浮点数转定点的关键核心是要理解浮点数(float、double)在ARM体系结构中的存储格式, 然后通过一个宏函数(用# define定义的一段代码)将其转化为定点数(INT, LONG LONG型数据)。不论是单、双精

收稿日期: 2009-05-15.
作者简介: 苏 扬(1963-), 男, 副教授. E-mail: su6303@163.com

度,两者都是基于 IEEE754 标准的浮点数存储格式。对于 IEEE754 单精度来讲[3],浮点格式共 32 位,包含 3 个构成字段:23 位小数 f,8 位偏置指数 e,1 位符号 S。将这些字段连续存放在一个 32 位字里,并对其进行编码。其中 0:22 位包含 23 位的小数 f;23:30 位包含 8 位指数 e;第 31 位包含符号 S,如图 1 所示。

IEEE754 双精度浮点格式(double)共 64 位,占 2 个连续 32 位字,包含 3 个构成字段:52 位的小数 f,11 位的偏置指数 e,1 位的符号位 S。将 2 个连续的 32 位字整体作为一个 64 位的字,进行重新编号。其中 0:51 位代表 52 位的小数,52:62 位代表 11 位的偏置指数,而最高位代表符号。

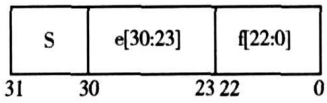

图1 单精度存储格式

根据浮点数的存储格式,就可以很快地写出一个浮点转定点的宏代码或内联函数(声明为 inline),而不是普通函数,因为函数调用的时候会发生参数的入栈出栈动作,这也需要耗费几个时钟周期,若此函数需要经常调用,则最好写成宏替代形式,减少这些不必要的时间浪费。当然转换后的定点数也要有相应的要求,一般定点数的格式是 32 位的(数据类型一般为 int),低 16 位代表小数,高 16 代表整数;若 2 个定点数相乘,其中 2 个 16 位小数相和其结果带有 32 位小数,这时还需要将结果右移 16 位,才最终得出乘法的结果,其流程图如图 2 所示。

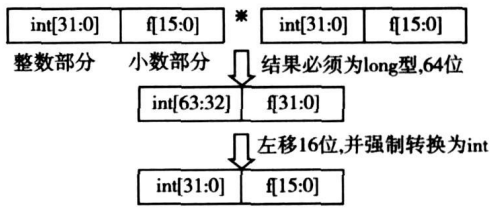

图2 32位定点数相乘过程处理

1.2 快速除法的实现

考虑用移位和乘法来代替除法,这是通常除法快速实现的优化思路,一次 32 位的乘法或乘累加操作最多需要 6 个周期。如果可以用乘法代替除法而且保持同样的准确度,可以大大提高计算的速度。该次使用的除法优化算法主要是基于以下表达式变型:$a/b = a \times (1/b) = a \times c$,只要有足够的精度来表示 $1/b$、a/b 与 $a \times c$ 的值可认为是相同的。

在 FFMPEG 工程中,WMA 解码中,特别是比特流操作中,有很多涉及诸如 pos/=2,pos/=8 等操作,遇到这些除数是 2 的幂数时,可将其转换为移位运算,即 pos>>=1,pos>>=3 等,这技巧可节省大量的时钟周期。

1.3 MDCT 阶段的特殊函数的优化

在 WMA 的解码环节当中,改进型离散余弦变换(MDCT)占了相当大的比重,其中大量地运用了 sin、cos、sqrt 等函数。直接运用 C 库里面的函数运算,至少需要几百个周期,对解码的速度影响很大。所以有必要改进这些函数功能的算法。在高等数学里面有关于 $\sin x$、$\cos x$ 的泰勒级数展开式[4],可以把 $\sin x$ 的求值写成如下等式

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} + \cdots + (-1)^m \frac{x^{2m+1}}{(2m+1)!} + \frac{\sin(m\pi + \pi + \theta x)x^{2m+2}}{(2m+2)!}$$

根据以上的泰勒展开式,可以很容易地将复杂 C 库里面的 $\sin x$ 算法转化为相对简单的几个乘除加法运算,先将 $\sin x$ 的 x 值转为定点形式的数据,再取泰勒级数的前 m 项(一般取前 6 项就足够了),后面的由于除法阶层比较大,对结果的精度影响不大,最终得出来的就是 $\sin x$ 的值,即 $\sin x$ 约等于前面 6 项。

$$\sin x \approx x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} - \frac{x^{11}}{11!}$$

同理,余弦 $\cos x$ 泰勒级数为

$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \cdots + (-1)^m \frac{x^{2m}}{(2m)!} + \frac{\cos(m\pi + \pi + \theta x)x^{2m+2}}{(2m+2)!},\ (0 < \theta < 1)$$

所以 $\cos x$ 的简单处理如下所示

$$\cos x \approx 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \frac{x^8}{8!} - \frac{x^{10}}{10!}$$

根据以上的 $\sin x$ 和 $\cos x$ 简化等式,用 c 代码就可以很简单地表达出来,如此可以节省数百周期。

1.4 汇编级的优化

通常在 ARM 平台上做汇编级的优化时,建议使用多数据批量装载和存储指令 LDM/STM,以减少总的执行周期数。从代码量来说,使用 LDM/STM 可以提高代码密度,有利于增加 Cache 命中率,理解上述

ARM9 装载/ 存储指令的特点有助于在程序优化上正确选择合适的指令。编写代码时可以调整指令的顺序, 尽量避免在装载指令后的 2 个周期内使用装载值, 流水线的效率就会比较高了。

ARM926EJ 内核支持单周期的 16 × 16 和 32 × 16 乘累加(MAC)操作、饱和算术指令以及前导零计数[5]。表 1 是 ARM9TDMI 和 ARM926EJ 乘累加操作的时间比较。

表 **1**　处理器的乘累加时间

处理器	带 32 位累加的 16× 16 乘法(cycles)	带 64 位累加的 32× 32 乘法(cycles)
ARMTDMI(ARMV4T)	4	7
ARM926EJ(ARMV5TE)	1	3

ARM926 的DSP 增强指令中的SMULAxy, 是带32 位累加的 16× 16 有符号数乘法指令, 如果在接下来的指令不立即用到结果, 只需要 1 个执行周期。因此在 MDCT 的计算中使用该指令来代替原来的 MLA 指令, 可以使乘法的速度提高到原来的 4 倍。相对来说, ARM926 对于音频应用更加适合, 因为音频数据一般是16 位精度的, 中间过程需要更多的精度, 在音频处理中 16 × 16 的乘累加运算非常多, 可以很好地利用 ARM926EJ 的乘法器特性。

2　FFMPEG 解码优化

FFMPEG 项目中有 wmadata. h、wmadec. c 2 个文件, 其中 wmadata. h 文件主要包含解码过程用到的静态数据, 无任何改进的地方; wmadec. c 源代码则描述 WMA 解码算法过程, 里面有大量的浮点类型数据, 因此上述的优化措施重点放在这个文件上。

首先, 将源代码中的所有 float 类型改为定点整数类型, 即如下定义:

```
# define fixed32        int32_t
# define fixed64        int64_t
```

根据上面浮点转定点原理, fixed32 和 fixed64 的高一半位代表整数部分, 低一半位代表小数部分, 其中 fixed64 主要用于定义计算结果类型, 这是为了提高计算的精度和防止数据溢出; 还有在 WMADecodeContext 这个解码关键数据结构当中, 须将 lsp_pow_e_table 等最后几个成员定义为 fixed64 类型, 因为它们会保存解码中间变量, 不能因为由于定点计算的移位操作而造成精度损失。

其次, 在 wma_decode_init( ) 函数当中, 有许多固定浮点系数, 这可直接转化为定点数并将其转为定点计算形式; 还有一些除数, 如 8、16、32 等这些以 2 为低的幂数, 这可以很容易地写为> > 1、> > 2 等形式, 大大缩减除法所用的周期数; 若遇到不是 2 的幂数, 根据快速除法的实现原理, 先实现计算倒数数值, 再转为定点数; 在 ff_imdct_calc( ) 这个改进离散余弦逆变换函数中, 有相当多的涉及正弦余弦变换的地方, 按照此类函数优化原则, 先写一个简单的静态处理函数, 再替代原来的 $\sin x$ 或 $\cos x$。

进一步, 在 wma_lsp_to_curve 函数中, 有 2 个 for 循环的计算, 这里的计算数据都是 16 位的, 利用 ARM926 的 32× 16 乘累加(MAC) 操作, 可以嵌入宏汇编计算函数, 从而进一步减少计算周期, 参考代码如下所示:

```
# define FIXMUL16( x, y)  \
({ int_hi, _lo, _result;  \
    asm ("MAC %0, %1, %3, %4\n\t"  \
        "movs %0, %0, lsr %5\n\t" "adc %2, %0, %1, lsl %6"  \
        :"= &r"(_lo), "= &r"(_hi), "= r"(_result)  \
        :"%r"(x),"r"(y),"M"(SIXTEEN),"M"(32-SIXTEEN)  \
        :"cc"); _result; })
```

最后要注意的是, 字节对齐问题, 因为ARM 是 4 字节对齐的, 所以在遇到将 unsigned char 类型地址转为 int 类型地址时, 要确保 char 型地址首先是字对齐的, 若不是则需要进行相应的移位确保对齐, 否则解码读取的数据序列会发生紊乱。

3 优化结果

经过上述优化后,直接的测试方法是在 I. MX21 (ARM926-EJ) 平台上以几个不同的采样率的 WMA 文件进行播放,文件的声道数、采样率、总帧数和每秒播放帧数等相关信息可以用 mplayer 播放软件通过串口打印得到,其中每秒帧率需要在 mplayer 的 main.c 主函数中自行设置一个打印函数才能取得,最终的测试结果经整理如表 2 所示。

表 2 解码数率对比

单频序列	解码速度(CPU 频率 213.81MIPS) 优化前\ 优化后
lalalove 44.1 k/s, 单声道, 3 250 帧	11 帧/s \ 26 帧/s
Always 32 k/s, 双声道, 4 390 帧	13 帧/s \ 28 帧/s
告别时刻 22 k/s, 双声道, 2 645 帧	15 帧/s \ 32 帧/s

从表 2 中的数据可以看出,优化后可以满足音频文件的实时播放速率。其中"解码速率"是解码这个音频序列的平均速度。这里的测试是在通过网络 NFS 方式进行的,即音频流还是在宿主机上,通过网络通信到目标板上,由于 CPU 需要处理额外的通信事务,解码速率和播放速率都会有所下降。

4 结 语

为使嵌入式 ARM926 平台上实现 WMA 多媒体文件的实时播放,以开源 FFMPEG 项目 WMA 解码部分为基础,从 4 个方面对 WMA 解码进行优化,使低端的嵌入式处理芯片也能较好地播放 WMA 文件。其中浮点转定点是优化的关键,也是优化工作的基础。当然针对 WMA 解码部分,还有其他地方可进行优化,今后研究的重点是解码中 MDCT 的快速算法实现优化,类似于 MPEG4 视频解码当中的 IDCT(反离散余弦变换),IDCT 有很多种快速算法,如 Loeffler 算法、ChenWang 算法,以普遍用于 MPEG4 的解码系统中,对加快视频流的解码有很大帮助,预计若对 MDCT 改进算法后,WMA 的解码速度会有进一步的提高。

参考文献

[1] FrankVahi. 嵌入式系统设计[M]. 骆 丽,译. 北京:北京航空航天大学出版社,2004.
[2] 丁贵广,郭宝龙,陈龙谭. 优化截断点的嵌入零块编码算法[J]. 西安电子科技大学学报:自然科学版,2003,30(2):195-207.
[3] 精英科技. 视频压缩与音频编码技术[M]. 北京:中国电力出版社,2001.
[4] 蔡安妮,孙景鳌. 多媒体通信技术基础[M]. 北京:电子工业出版社,2000.
[5] 杜春雷. ARM 体系结构与编程[M]. 北京:清华大学出版社,2002.